SymbolicModelCheckingforConverse Propositional Dynamic Logicbased Industry 4.0

Leelavathi Rajamanicham[1]

School of Information Technology SEGi University Malaysia.

Qin FenPing[2]

School of Information Technology, SEGi University Malaysia Campus of Nanning GuilingUniversity of Technology China

Chen Jia[3]

School of Information Technology, SEGi University Malaysia Campus of Nanning GuilingUniversity of Technology China

Abstract:*Propositional Dynamic Logic with Converse (CPDL) is a modal logic developed for reasoning about programs with applications for instance in knowledge representation. This paper presents a symbol model checking algorithm for propositional dynamic logic with converse. First of all, according to the analysis of symbolic model checking technology by ordered binary decision diagram, we describe how to represent propositional dynamic logic with converse model system symbolically using ordered binary decision diagram. Then a symbolic model checking algorithm for propositional dynamic logic with converse is proposed. Finally, the proof of the algorithm is given, and the validity of this algorithm was verified by an example.*

Keywords:*model checking; Converse propositional dynamic logic(CPDL); Ordered Binary Decision Diagram (OBDD)*

I. INTRODUCTION

Propositional Dynamic Logic (PDL) [1] is best used to describe the state attributes that a program reaches during execution and to simulate the evolution of the computational process. Over the years, propositional dynamic logic has proven to be a valuable theoretical tool for computer science, logic, computational linguistics, and artificial intelligence. The decidability and complexity of many reasoning programs depend on the research in the propositional dynamic logic.Converse Propositional Dynamic Logic (CPDL), also defined in [2], extends propositional dynamic logic with a converse operation on programs.Both propositional dynamic logic and converse propositional dynamic logic have the small-model property and are EXPTIME-complete.

The research on the dynamic logic of proposition is divided into the research of the nature and function of proposition dynamic logic and its extended propositional dynamic logic [3][4], and the application research of propositional dynamic logic. Application examples are: program verification [5], Semantic Web [6], describing dynamic evolution based on agent system [7], planning [8], knowledge engineering [9]. It is also related to cognitive logic [10] and closely related to description logic [11]. There is also research on the proposition dynamic logic decision algorithm and algorithm efficiency [12] [13].

The initial implementation of the model checking algorithm is to use the associated linked list to represent the formula and migration relationship. In practical applications, only models with a small number of states can be detected. However, for models with more states, this method of describing the model with an associated linked list cannot be processed because the state transition diagram is too large. Ordered Binary Decision Diagrams (OBDD) [14] [15] can effectively represent the transition between finite state machines and states, and can represent a set of transitions between states and states rather than individual states. In the 1990s, the symbolic model checking technique based on ordered binary decision diagram proposed by McMillian et al. [15] greatly improved the detection ability of model checking. Symbolization technology enables model checking to detect more states, enabling model checking techniques to be widely used in practical applications.

In view of the efficiency of ordered binary decision diagram, branching time Computation Tree temporal Logic (CTL), and Linear Time Temporal Logic (LTL) use ordered binary decision diagram to symbolize the model checking algorithm to improve the efficiency of the algorithm. The good application of model checking technology in the field of planning has been recognized by many experts and scholars. The research on various symbolic model checking algorithms based on temporal logic framework has also achieved good results. Although temporal logic language has a strong ability to describe, it does not accurately describe certain characteristics of industrial production. Propositional dynamic logic is a kind of logical language that focuses on state changes caused by actions. This feature is similar to the changes in the execution of various operations in industrial production. Therefore, we consider using propositional dynamic logic as the logical basis of the symbolic model checking technique, researching the model checking technology of propositional dynamic logic, and presenting the converse proposition dynamic logic symbol model checking algorithm based on ordered binary decision diagram.

II. BASIC CONCEPTS

A. Converse Propositional Dynamic Logic

The regular proposition dynamic logic has operators that represent the sequential connection, selection, testing and iteration of the action. This paper expands on the basis of the regular proposition dynamic logic language, extends the Converse action operator, and calls the extended logic language the propositional dynamic logic with converse.

Definition 1 CPDL syntax Let Φ_0 and $\prod_0$ denote the atomic proposition set and atomic program set in in the converse propositional dynamic logic respectively, φ and ψ denote propositions, and α and β denote programs. The proposition set Φ and the program set $\prod$ are recursively defined as follows： $\Phi_0 \subseteq \Phi$

(1) $\prod_0 \subseteq \prod$

(2) If φ、$\psi \in \Phi$, then $\varphi \to \psi \in \Phi$, $0 \in \Phi$, $\varphi \wedge \psi \in \Phi$, $\neg\varphi \in \Phi$

(3) If α、$\beta \in \prod$，then α; $\beta \in \prod$，$\alpha \cup \beta \in \prod$，$\alpha^* \in \prod$，$\alpha^- \in \prod$

(4) If $\varphi \in \Phi$ and $\alpha \in \prod$,then $[\alpha]\varphi \in \Phi$

(5) If $\varphi \in \Phi$, then $\varphi ? \in \prod$

Definition 2CPDLsemantic The converse propositional dynamic logic semantic interpretation is based on a Kripke structure of a binary group $\Re=(Kim)$ Where K is a set of states, m_{ake} represents a set of states or binary relationships, Φrepresents a set of propositions, $\prod$ represents a set of programs, and

$$m_K(\varphi) \subseteq K, \qquad \varphi \in \Phi$$

$$m_K(\alpha) \subseteq K \times K, \qquad \alpha \in \Pi$$

Thesemantic interpretation of the converse propositional dynamic logic is as follows:

$$m_K(\neg\varphi) = K - m_K(\varphi)$$

$$m_K(\varphi \to \psi) = (K - m_K(\varphi)) \cup m_K(\psi)$$

$$m_K([\alpha]\varphi) = K - (m_K(\alpha) \circ (K - m_K(\alpha)))$$

$= \{u \mid \forall v \in K, if\,(u,v) \in m_K(\alpha) then\, v \in m_K(\varphi)\}$ $m_K(\varphi \wedge \phi) = m_K(\varphi) \bigcap m_K(\phi)$

$m_K(\alpha^{-}) = \{(u,v) \mid (v,u) \in m_K(\alpha)\}$

$m_K(\alpha;\beta) = m_K(\alpha) \circ m_K(\beta)$

$= \{(u,v) \mid \exists w \in K, (u,w) \in m_K(\alpha) and (w,v) \in m_K(\beta)\}$ $m_K(\alpha \cup \beta) = m_K(\alpha) \cup m_K(\beta)$

$m_K(\alpha^{*}) = m_K(\alpha)^{*} = \bigcup_{n \geq 0} m_K(\alpha)^{n}$

$m_K(\varphi?) = \{(u,u) \mid u \in m_K(\varphi)\}$

Where the symbol $\circ$ represents the synthesis of the relationship

For any state $v \in K$, when $v \in m_{ake}(\varphi)$, it is usually denoted as $\Re, v \vDash \varphi$, and it is said that the state v satisfies φ in the model $\Re$, or the state v of φ in the model $\Re$ is true. When the model $\Re$ is obvious, $\Re$ is often omitted, denoted as $v \vDash \varphi$. Conversely, when $v \notin_K (\varphi)$, it means that the state v does not satisfy φ.

B. Ordered Binary Decision Diagram (OBDD)

Ordered binary decision diagram, this symbolic data structure is an expression of a Boolean function, so it must also be able to perform the corresponding Boolean operations by manipulating the ordered binary decision diagram. In fact, the symbolic operations of many Boolean functions can also be implemented by a graphical algorithm of an ordered binary decision diagram.

Definition 3 The binary decision diagram is a directed acyclic graph $G(V \cup T, E)$ for representing a cluster of Boolean functions f_i: $\{0,1\}^n \rightarrow \{0,1\}$ based on variables $x_1, x_2, \ldots, x_n$, which satisfies:

(1) Each node in the diagram corresponds to a unique function f_i

(2) A node without an edge is called an endpoint, and there are only two endpoints, 0 and 1. The set of endpoints is denoted as T.

(3) The other nodes except the endpoint are called inner nodes, and the set of inner nodes is V. For $\forall v \in V$, it is identified by the variable name $var(v)$ and has two exit edges. The exit edge after v is 0 is called 0-edge, and the exit edge after v is 1 is called 1-edge. The set of all sides is denoted as E. The nodes pointed by the 0-edge and the 1-edge are denoted as $low(v)$ and $high(v)$, respectively.

(4) On the directional path of the binary decision diagram, each variable appears at most once.

Definition 4 A Binary Decision Diagram is an Ordered Binary Decision Diagram (OBDD) G (V, E). If there is a full order relationship < on the variable set, the following conditions are satisfied: for any non-terminal $u\in V$, If $v\in V$ is the node pointed to by the exit edge of u, and v is also a non-endpoint, then there must be $var(u)<var(v)$.

Definition 5 The node v on each ordered binary decision diagram represents a Boolean function $f(v)$:$\{0, 1\}^n\rightarrow\{0,1\}$, and satisfies:

(1) If v is the end point, $f(v)$ represents a constant function whose value is the value of the end point v.

(2) If v is a non-terminal point, then $f(v)=\mathrm{var}(v)\cdot f(high(v))+\overline{\mathrm{var}(v)}\cdot f(low(v))$. Where "·" means logical multiplication, "+" means logical addition, and "-" means logical complement.

Theorem 1[14]For Boolean functions $f(x_1,x_2,...,x_n)$ from $\{0,1\}^n$ to $\{0,1\}$ and a given variable order, there is a unique reduced ordered binary decision diagram representation of the Boolean function, that is, the ordered binary decision diagram under the variable order is the standard form of Boolean functions $f(x_1,x_2,...,x_n)$.

III. ORDERD BINARY DECISION DIAGRAMREPRESENTATIONOFCONVERSE PROPOSITIONAL DYNAMIC LOGICMODEL

Propositional dynamic logic with converse is semantically interpreted on the Kripke structure. In addition to using Boolean strings to represent states and propositions in the converse propositional dynamic logic migration model, you also need to represent the action.

If the state set ***S*** of the system has a total of m states, the state is represented by an n-bit binary string, and the determination of n satisfies the relationship: $2^{n-1}<m\leq 2^n$. If the set S has a subset T, then a Boolean function f_T is defined: $\{0,1\}^n\rightarrow\{0,1\}$. If the state is $t\in T(t\in \boldsymbol{S})$, the Boolean function f_T maps the Boolean string $[v_1v_2...v_n]$ of t to 1, otherwise it maps to 0. Therefore, a Boolean function f_T can be used to represent any subset of S Its Boolean characteristic function is $f=x_1x_2\ldots x_i\ldots x_n$, where x_i is appearing when v_i=1, and when v_i= 0, x_i appears negatively in the form of $\overline{x_i}$. From the semantic interpretation of propositions in propositional dynamic logic with converse, the interpretation function $m_{\mathfrak{R}}$ maps the propositions into a set of states, i.e.$m_{\mathfrak{R}}(\varphi)\subseteq S$, $\varphi\in\Phi$. Thus a subset of the state set ***S*** can be used to represent the proposition.

Given a Kripke structural model of converse propositional dynamic logic:

$\boldsymbol{S}=\{s_0, s_1, s_2\}$

$m_{\mathfrak{R}}(p)=\{s_0, s_2\}$

$m_{\mathfrak{R}}(q) = \{s_1, s_2\}$

$m_{\mathfrak{R}}(a) = \{(s_0, s_0), (s_0, s_1)\}$

$m_{\mathfrak{R}}(b) = \{(s_1, s_1), (s_0, s_2), (s_2, s_1)\}$

The state transition diagram of the model is shown in Fig 1.

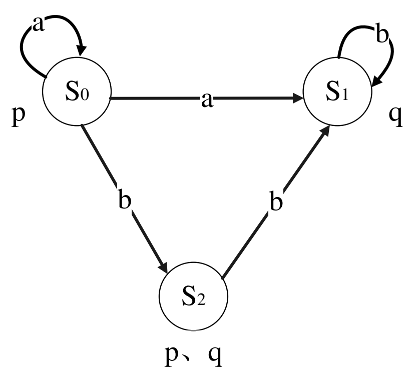


Fig 1 converse propositional dynamic logic migration model

According to the state's ordered binary decision diagram representation rule, a Boolean expression with a length of 2 bits can be used to represent the state, and the proposition is represented by a set of states that satisfy the proposition, which is specifically expressed as follows:

Status	Boolean function	Boolean expression
s_0	$\overline{x}_1\overline{x}_2$	00
s_1	$\overline{x}_1x_2$	01
s_2	$x_1\overline{x}_2$	10

Proposition	Boolean function	Boolean expression
p	$\overline{x}_1\overline{x}_2 + x_1\overline{x}_2$	00+10
q	$\overline{x}_1x_2 + x_1\overline{x}_2$	01+10

The corresponding ordered binary decision diagram representation can be obtained from the Boolean expression.

From the semantic interpretation of the action in the converse propositional dynamic logic, the interpretation function m_K maps the action to a state pair: $m_K(\alpha)\subseteq S\times S$, $\alpha\in\prod$. Therefore, the action can be represented by two states before and after the migration execution, that is, the action is represented by a

2n-bit binary string, where the first n bits are the precursor state s of the action, and the last n bits are the successor state s' that arrives after the action is executed. It is like the state pair $[(v_1v_2...v_n)\ (v'_1v'_2...v'_n)]$. If an action consists of multiple transitions, the action is represented by a set of all migrated state pairs.

Action a in a given model consists of 2 transitions, then m_K (α)= $\{(s_0\ s_0), (s_0\ s_1)\}$. According to the representation method of the set in the ordered binary decision diagram, the Boolean functions of action a and action b are respectively:

$f_{\mathrm{a}}= \overline{x}_1\overline{x}_2\ \overline{x}_1'\overline{x}_2'+ \overline{x}_1\overline{x}_2\overline{x}_1'x_2'$

$f_{\mathrm{b}}=\overline{x}_1\overline{x}_2\ x_1'\overline{x}_2'+ x_1\overline{x}_2\ \overline{x}_1'x_2'+\overline{x}_1x_2\overline{x}_1'x_2'$

The Boolean expression can be used to obtain the corresponding ordered binary decision diagram representation of the action

IV. CONVERSEPROPOSITIONALDYNAMIC LOGICSYMBOLIC MODEL CHECKING ALGORITHM

In the model checking system, usually the given system specification $\mathcal{F}$ is a formula of the form s⊨<π>Φ. Through the semantic interpretation of the converse propositional dynamic logic and the corresponding operations in the ordered binary decision diagram, the action sequence π and the proposition set Φ can be simplified to the simplest form.

Below we present a converse propositional dynamic logic model checking algorithm based on ordered binary decision diagram.

Algorithm 1 For the Kripke structure $\mathfrak{K}$ and the converse propositional dynamic logic formula $\mathscr{F}$ for a given system to be tested, model checking by the following steps:

(1) A symbolic representation of the action is further obtained by binary encoding of the state, wherein the action is represented by a state pair consisting of the state before and after the action is executed.

(2) The converse propositional dynamic logic formula $\mathscr{F}$ of the form <π>Φ,through the Get BDD π and Get BDDΦ algorithms respectively transforms the action and proposition into the simplest symbolized representation.

(3) . For the BDD π and BDD Φ obtained in the previous step, by the exist algorithm, a symbolized representation of all states satisfying the property Φ after the action sequence π can be obtained. A state set $\mathbf{S}_{\mathscr{F}}$ satisfying the formula $\mathscr{F}$ can be finally obtained by analysis.

(4) Get model checking results. It is judged that $S\cap \mathbf{S}_{\mathscr{F}}=\mathbf{\emptyset?}$, if not empty, the result "$\mathscr{F}$ is satisfied with respect to the model $\Re$"; otherwise, "$\mathscr{F}$ is unsatisfiable with respect to the model $\Re$".

Theorem 2 Given a Kripke structure $\Re= (K, mK)$, for any proposition dynamic logic with converse formula φ. If φ is a proposition, then s∈BDD(φ) established, if and only if $\Re$,s⊨φ; if φ is a program, then (s,t)∈BDD(φ), if and only if (s,t)= m_K(φ).

Proof: Suppose there are states :s, t, v∈K, the situation is considered as follows

(1) φ is a proposition:

When φ is an atomic proposition, s∈BDD(φ) denotes that s is an element in the set of states satisfying the proposition φ, and obviously R, s⊨φ holds. On the contrary, $\Re$,s⊨φ denotes that the formula φ is satisfiable at the state s, and it is apparent that the state s is one of the elements in the state set satisfying the formula φ, that is, s∈BDD(φ) is established. The same can be proved: $\varphi=\varphi_1\vee\varphi_2$ 和 $\varphi=\neg\varphi_1$.

When $\varphi=<\alpha>\varphi_1$, according to the semantics of $<\alpha>\varphi_1$, BDD(<α>φ1) is an ordered binary decision diagram set composed of all the precursor states of those state pairs satisfying the migration relationship α and whose successor states satisfy φ1. s∈ BDD($<\alpha>\varphi_1$), that is, s is one of the states satisfying $<\alpha>\varphi_1$, so that $\Re$,s⊨φ holds. On the contrary, $\Re$,s ⊨$<\alpha>\varphi_1$ denotes that s is one of the states satisfying the formula $<\alpha>\varphi_1$, and BDD($<\alpha>\varphi_1$) is an ordered binary decision diagram composed of all states satisfying $<\alpha>\varphi_1$. Therefore, there is s∈ BDD($<\alpha>\varphi_1$).

(2) φ is the action:

When φ is an atomic action, BDD(φ) constitutes an ordered binary decision diagram from all state pairs satisfying action φ, and (s,t)∈BDD(φ), that is, state pairs (s, t) are all satisfying BDD(φ) state of one of the set, according to its semantic interpretation, obviously there is (s,t)∈m_K(φ) . On the contrary, (s,t)∈m_K(φ), that is, the state pair (s,t) satisfies the semantic interpretation of the action φ, that is, there is a transition relationship between the states s and t, and BDD(φ) is composed of all states satisfying the action φ, it is apparent that (s,t)∈BDD(φ) holds. The same can be proved: $\varphi=\varphi_1\cup\varphi_2$ and $\varphi=\varphi_1^-$.

When$\varphi=\varphi_1;\varphi_2$, BDD($\varphi_1;\varphi_2$) is an ordered binary decision diagram composed of state pairs (s, t), this state pair consists of state pair (s,v) satisfies φ_1 and state pair (v,t) satisfies φ_2 . Therefore, the state pair (s, t) satisfies the semantic solution m_K ($\varphi_1;\varphi_2$) of $\varphi_1;\varphi_2$.that is, (s,t)∈m_K ($\varphi_1;\varphi_2$) holds. On the contrary, the state pair (s,t)∈m_K ($\varphi_1;\varphi_2$) indicates that(s,t) satisfies$\varphi_1;\varphi_2$, and BDD($\varphi_1;\varphi_2$) is composed of all pairs of states satisfying$\varphi_1;\varphi_2$, and therefore $(s,t) \in m_K(\varphi_1;\varphi_2)$.

When $\varphi=\varphi_1?$,checks whether the current state satisfies φ_1, so if $s \in BDD(\varphi_1?)$, that is, s satisfies φ_1, then $\Re,s \vDash \varphi_1?$ holds. On the contrary, $\Re,s \vDash \varphi_1?$ that is, the s state satisfies φ_1, and thus $s \in BDD(\varphi_1?)$.

When$\varphi=\varphi_1^*$,φ_1^* is the reachability relationship of all the models satisfying the formula φ. According to the semantics and the composition of ordered binary decision diagram, the theorem is also true when$\varphi=\varphi_1^*$.

Ordered binary decision diagram is a canonical representation of Boolean expressions. Boolean operations (OR, AND, NAND, XOR, etc.) on ordered binary decision diagram are polynomial time, and equivalence judgments are constant time. In view of the one-to-one correspondence between the ordered binary decision diagram and the Boolean function, the Boolean function operation is used to illustrate the model checking process.

Example 1 First, enter the formula to be verified :$f=\langle a^-;b\rangle q$, then

OBDD f= exist (BDD (a^-;b), BDDq)

OBDD (a) $\leftrightarrow f_a= \bar{x}_1\bar{x}_2\bar{x}_1'\bar{x}_2'+\bar{x}_1\bar{x}_2\bar{x}_1'x_2'$

OBDD (a^-) $\leftrightarrow f_{a^-} = \bar{x}_1\bar{x}_2\bar{x}_1'\bar{x}_2'+\bar{x}_1 x_2\bar{x}_1'\bar{x}_2'$

OBDD (b) $\leftrightarrow f_b=\bar{x}_1\bar{x}_2\bar{x}_1'x_2'+x_1\bar{x}_2 x_1'\bar{x}_2'+x_1\bar{x}_2\bar{x}_1'x_2'$

Introducing operations on Boolean functions $B(f,(x\rightarrow y))$, This operation replaces the variable x in the Boolean expression f with the variable y.

$B(f_{a^-}, (x'\rightarrow y))= \bar{x}_1\bar{x}_2\bar{y}_1\bar{y}_2+\bar{x}_1 x_2\bar{y}_1\bar{y}_2$

$B(f_b, (x\rightarrow y))=\bar{y}_1\bar{y}_2 x_1\bar{x}_2'+y_1\bar{y}_2\bar{x}_1'x_2'+\bar{y}_1 y_2\bar{x}_1'x_2'$

$f_{(a^-;b)} = B(f_{a^-}, (x'\rightarrow y)) \wedge B(f_b, (x\rightarrow y))$

$=(\bar{x}_1\bar{x}_2\bar{y}_1\bar{y}_2+\bar{x}_1 x_2\bar{y}_1\bar{y}_2)\wedge(\bar{y}_1\bar{y}_2 x_1\bar{x}_2'+y_1\bar{y}_2\bar{x}_1'x_2+\bar{y}_1 y_2\bar{x}_1'x_2')=\bar{x}_1\bar{x}_2\bar{y}_1\bar{y}_2 x_1\bar{x}_2'+\bar{x}_1 x_2\bar{y}_1\bar{y}_2 x_1\bar{x}_2'$

After eliminating the intermediate variable y

$f_{(a^-;b)}=\bar{x}_1\bar{x}_2 x_1\bar{x}_2'+\bar{x}_1 x_2 x_1\bar{x}_2'$

$B(f_q, (x\rightarrow x')) = \bar{x}_1 x_2 + x_1\bar{x}_2=\bar{x}_1'x_2'+x_1\bar{x}_2'$

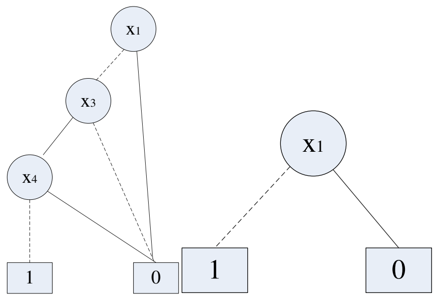
(BDD(a^-;b)& BDDq)- BDDq$\leftrightarrow(f_{(a^-;b)}\wedge f_q)-f_q$

$=(\overline{x}_1\overline{x}_2x_1{'}\overline{x}_2{'}+\overline{x}_1x_2x_1{'}\overline{x}_2{'}) \wedge (\overline{x}_1{'}x_2{'}+x_1{'}\overline{x}_2{'})-(\overline{x}_1{'}x_2{'}+x_1{'}\overline{x}_2{'})$

$=(\overline{x}_1\overline{x}_2x_1{'}\overline{x}_2{'}+\overline{x}_1x_2x_1{'}\overline{x}_2{'})-(\overline{x}_1{'}x_2{'}+x_1{'}\overline{x}_2{'})$

$=\overline{x}_1\overline{x}_2+\overline{x}_1x_2$

Through the above calculation, find a set of all states that satisfy the formula f =<a¯ ;b>q, that is $\{s_0,s_1\}$. Since $\{s_0\}\cap\{s_0,s_1\}\neq\emptyset$, it is apparent that the state s_0 is a subset of the state set satisfying the formula f. Therefore, in the model K, the formula K, $s_0\vDash$<a¯ ;b>q is established. Among them, the ordered binary decision diagram of a¯ ;b and <a¯ ;b>q is shown in Fig. 2.



(1) a¯ ;b (2) <a¯ ;b>q

Fig 2: OBDD representation of the formula f =<a¯ ;b>q

V. CONCLUSION

Many practical problems can be formulated with propositional dynamic logic, and the solution to the problem is based on the appropriate framework for reasoning. The proposition dynamic logic model checking is another way to solve the problem solution, and it is also a new application of proposition dynamic logic. In this paper, the algorithm of proposition dynamic logic with converse symbol model detection based on ordered binary decision diagram is given, and the algorithm is proved to be correct and feasible by examples. Applying other model checking methods such as boundary model checking to propositional dynamic logic to improve proposition dynamic logic model checking method is a further research content.

REFERENCES

[1] D. Harel, D. Kozen and J. Tiuryn.(2000). Dynamic Logic, *MIT Press,* Cambridge, MA.

[2] M. Fischer and R.Ladner. (1979). Propositional dynamic logic of regular programs.*Journal of Computer and System Sciences, 18(2):194-211.*

[3] C.Lutz. (2005).PDL with Intersection and Converse is Decidable.In *Annual Conference of the*

European Association for Computer Science Logic CSL'05, LNCS. *Springer Verlag.*

[4] Stefan Göller, Markus Lohrey, Carsten Lutz.(2007). PDL with Intersection and Converse is 2EXP-complete. Algorithmic-Logical Theory of Infinite Structures.

[5] Benevides, Mario R. F.(2008). A propositional dynamic logic for CCS programs.*In15th International Workshop on Logic, Language, Information and Computation, Springer Verlag*

[6] Fahima Cheikh, Giuseppe De Giacomo.(2006).

Automatic Web Services Composition in Trust-aware Communities. *ACM*

[7] J.C. Meyer, Dynamic logic reasoning about actions and agents.(1999). *In Proc. Workshop on Logic-Based Arti-ficial Intelligence, Washington, DC, USA*

[8] L. Spalazzi, P. Traverso. (2000). A dynamic logic for acting, sensing, and planning, *J. Logic Comput. 10 (6): 787–821*

[9] F. van Harmelen, J. Balder.(1992). A formal language for KADS models of expertise.*Knowledge Acqui-sition 4: 127–161*

[10] H.P. van Ditmarsch, W. van der Hoek, B.P. Kooi. (2003). Concurrent dynamic epistemic logic for MAS. *In Proc.2nd Int. Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia, ACM Press, pages 201–208*

[11] G.D. Giacomo, M. Lenzerini.(1994). Boosting the correspondence between description logics and propositional dynamic logics. *In Proc. of the 12th National Conference on Artificial Intelligence, AAAI '94, AAAI Press/MIT Press, pages 205–212.*

[12] GiuseppeDeGiacomo,Fabio Massacci. (2000) .Combining Deduction and Model Checking into Tableaux and Algorithms for Converse-PDL. *Inf. Comput. 162(1-2): 117-137.*

[13] M. Lange. (2006). Modelchecking propositional dynamic logic with all extras. *Journal of Applied Logic, 4:39-49.*

[14] Bryant R. (1986).Graph-Base Algorithm for Boolean Function Manipulation.*Transactions on computer , pages 677-691*

[15] BRYANT R E. (1992). Symbolic Boolean manipulation with ordered binarydecision diagrams. *ACM Computing Surveys,124 (3):293-318*

[16] E. Clarke, Orna Grumberg and D. Peled.(2000) Model Checking,*MIT Press*

[17] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J.(1990). Hwang. Symbolic model checking: 10^{20} states and beyond. *In Proc. Symposium on Logic in Computer Science, pages 428–439. IEEE.*

[18] Stefan Göller, Markus Lohrey' and Carsten Lutz. (2009).PDL with intersection and converse: satisfiability and infinite-state model checking. *Volume 74, pages. 279-314.*

[19] B. Yang, R. E. Bryant, D. R. O'Hallaron, A. Biere, O. Coudert, G. Janssen, R. K. Ranjan & F. Somenzi.(1998).A Performance Study of BDD-Based Model Checking. *InProceedings of the Second International Conference on Formal Methods in Computer-Aided Design, FMCAD '98, Springer Verlag, London, UK, pages. 255–289.*

[20] A Goel, G. Hasteer & R. Bryant.(2003).Symbolic representation with ordered function templates. *InProceedings of the 40th annual Design Automation Conference*

[21] Christian Doczkal, Joachim Bard (2018) Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq

[22] Sedla´r, I. (2016).Non-classical PDL on the cheap. Manuscript.

[23] Teheux, B. (2014). Propositional dynamic logic for searching games with errors. *Journal of Applied Logic 12(4), 377–394*